Computer Science and Information Technology 5(2): 66-73, 2017 http://www.hrpub.org
DOI: 10.13189/csit.2017.050204

VUZALIZER: A Max/MSP Object for Real-time Generation of RCMC Canons

Alba Francesca Battista[1,*], Nicola Monopoli[2], Matteo Nicoletti[3]

[1]Department of New Technologies and Musical Languages, Conservatory «D. Cimarosa» of Avellino, Italy
[2]Department of New Technologies and Musical Languages, Conservatory «U. Giordano» of Foggia, Italy
[3]Independent Researcher, Italy

Abstract Regular Complementary Canons of Maximal Category (RCMC) or *Vuza Canons* were introduced to the musical world with Dan Tudor Vuza's seminal papers at the pnm Conference in the 1990s. Musicians have always been intrigued by canon construction, q.v. the complex polyphony of the Flemish composer Josquin Desprez or the contrapuntal techniques of Johann Sebastian Bach, whose properties have been translated into formal algebraic terms. Nowadays the process of building mosaic canons can be implemented using programming languages allowing composers to use these extremely complex macro-structures. In this paper we will discuss our algorithm that allows composers to create and directly manage RCMC Canons. In addition, we will describe VUZALIZER, a Max/MSP object which generates Vuza Canons.

Keywords Vuza Canons, RCMC, Max/MSP, Spatialization, Mathematica

1. Introduction

A canon is a recognizable pattern that is repeated with different offsets, typically with different instruments or simply different voices. This pattern, called *motif* or *inner voice*, can be modified (i. e. augmented or retrograded). In this paper, we will take into account the construction of a particular type of canon, the rhythmic one.

When we study *rhythmic* canons, we only need to consider their rhythmic properties (i.e. the occurrences of musical events, regardless of pitch, timbre or dynamics). A rhythmic profile acts as a *tile* that the composer tries to translate temporarily in order to construct macro-structures having the properties to fill the entire space of the pulse, without silence between a rhythmic pulsation and another, and without superpower position among the various voices, which are therefore mutually complementary. A rhythmic canon of this type is also called *mosaic* since it creates a regular tiling of a rhythmic space.

The geometric properties of these entities, after more than twenty years of research, are still a topic of great relevance for mathematicians and musicians.

Musicians have always held a fascination for musical canons, q.v. the complex polyphony of the Flemish composer Josquin Desprez or the contrapuntal techniques of Johann Sebastian Bach, whose properties have been translated into formal algebraic terms. Olivier Messiaen was maybe the first theorist and composer who introduced and studied the concept of rhythmic canon. In some of his compositions, for example *Visions de l'Amen* (1943) for two pianos or *Harawi* (1945) for piano and voice, the use of rhythmic canons anticipates some formal features of mosaic music, although he did not give a formalization of this compositional process.

From a mathematical point of view, the construction of rhythmic canons is formalized in terms of factorizations of finite cyclic groups as the sum of two subsets, as we will later observe a concept of which Messiaen, like many other composers of the Twentieth century, was probably unaware of. An exception is surely the composer and engineer Iannis Xenakis who used mathematical tools for the development and the formal organization of the compositional material.

Nowadays the situation is different thanks to the development of music informatics. In fact, the process of building mosaic canons can be implemented using programming languages allowing composers to access extremely complex macro-structures that can be taken as a basis for formal architectures of original musical compositions.

2. Rhythmic Canons

As we said, a canon is a contrapuntal compositional technique or texture that employs a melody with one or more imitations of the melody played after a given duration. If we

choose a motif A, transformations τ_i, $i \in I$, that may be only offsettings (viz. translations in time), a canon is the reunion C of the transforms $\bigcup_{i\in I}\tau_i(A)$. If a motif is identified with its characteristic function $1_A : D \to \{0,1\}$, with D which is a subset of integers Z, then the superimposition of all its copies appears as a sum: $C = \sum_i 1_{\tau_i(A)}$ [1].

In the most cases musical canons in the classical sense have nil to n notes for each beat. However, it is possible to create more interesting types of canons; we distinguish them in *coverings* and *packings*. The first type is a canon, in which each available beat features one or more notes, $(\forall t \in D, C(t) \geq 1)$. The second one is a canon where there is never more than one note for each possible beat $(\forall t \in D, C(t) \leq 1)$. Considering the entire problem from a mathematical point of view, it is interesting to define it requesting one and only one note per beat; this is called *tiling* (i.e. a mosaic with copies of one motif) eventually allowing deformations. We can write our problem as:

$$tilings = coverings \cap packings : \forall t \in D, C(t) = 1 \quad (1)$$

2.1. How to Define a Rhythmic Canon

Let G be a finite *cyclic group* [1], and S, R two non-empty subsets of G. If each element $g \in G$ can be expressed uniquely in the form $g = s + r$ with $s \in S$ and $r \in R$ then the equation

$$G = S \oplus R \quad (2)$$

is called a *factorization* of G [2]. A rhythmic canon of G is a factorization of G in two subsets. The subset S is called the *inner voice* or *motif*, while R is called the *outer voice*. If G is the cyclic group $Z_N = \{0,1,2,\dots,N-1\}$, the set S can be seen as a tile of the line Z_N and we said that S tiles Z_N with R or that R tiles with S, since if (S,R) is a rhythmic canon, also (R,S) is a rhythmic canon. We can say that tiling the line is the same as finding a decomposition of the generic cyclic group Z_N.

A non-empty set S is *periodic* if there is an element g of G such that $S + g = S$ and $g \neq 0$ (the identity element of G). In 1948, the Hungarian mathematician György Hajós hypothesized that in a factorization of Z_N in two factors, one of the factors had to be periodic. This conjecture is false, and the counterexamples are exactly Vuza Canons.

3. Vuza Canons

A factorization of a cyclic group is said to be trivial if at least one of its factors is periodic. In the 1950s, Hajós proved that it is not necessary for one of the two subsets to be periodic. From 1941 to 1957, mathematicians such as Nicolaas Govert de Bruijn, László Rédei and Hajós himself showed several examples of groups for which this is true.

In the 1990s, Dan Tudor Vuza published four articles devoted to the formalization of a particular class of rhythmic canons: RCMC canons [3].

Let's start with some definitions.

Definition 1. Let G be an Abelian group. A *k-factorization* of G is a factorization in the direct sum of k of its elements.

A *k-factorization* of $G = A_1 \oplus A_2 \oplus \dots \oplus A_k$ is *periodic* if exists an index $i \in \{1,2,\dots k\}$ so that A_i is periodic.

A *k-factorization* which is not periodic is said *aperiodic*.

Definition 2. Let G be an Abelian group.

G is *k-Hajós* if each of its *k-factorizations* is periodic.

G is *not k-Hajós* if exists at least one of its *k-factorization* which is aperiodic.

For $k = 2$, we will simply say *of Hajós* or *non-Hajós*.

Definition 3. Let G be a finite cyclic group and S, R two non-empty subsets of G. A *Vuza Canon* (S,R) is a rhythmic canon

$$G = S \oplus R \quad (3)$$

where neither S nor R is periodic. In other words, a Vuza canon is a *2-factorization* of a cyclic group. The order N of the group is the canon *period*.

Now, we are going to enounce some theorems that are necessary in order to implement the algorithm for the creation of RCMC Canons. The proof of each of them is given in many group theory textbooks (e.g. in papers by Brujin [4], Amiot [5] and Jedrzedjewski [6]).

Theorem 1. The following statements are equivalent:

(i) Vuza Canons only exist for orders N which are not in the form

$$p^\alpha(\alpha \geq 1), p^\alpha q(\alpha \geq 1), p^2q^2, p^\alpha qr(\alpha = 1,2), pqrs \quad (4)$$

where p, q, r, s are different primes.

(ii) Vuza Canons only exist for orders N of the form

$$N = n_1 n_2 n_3 p_1 p_2 \quad (5)$$

where p_1, p_2 denote different primes, $n_i p_i \geq 2$ for $i = 1,2$ and $n_1 p_1$ and $n_2 p_2$ are coprime.

This first theorem shows some of the essential conditions to find a period for a Vuza Canon.

Theorem 2. Let G be a finite cyclic group and S, R two non-empty subsets of G. The following statements are equivalent.

(i) The sum $S + R$ is direct and is equal to G.

(ii) $G = S + R$ and $|G| = |S||R|$.

Theorem 3. Let p_1, p_2 denote different prime numbers, and n_1, n_2, n_3 be positive integers such that the product $n_1 p_1$ is coprime with $n_2 p_2$, then the pair (S,R) defined by

$$S = A + B \quad (6)$$

$$R = (U + V' + K_1) \cup (U' + V + K_2) \quad (7)$$

is a Vuza Canon of Z_N with $N = n_1 n_2 n_3 p_1 p_2$ if R is a non-periodic subset of Z_N and:

1 A *cyclic group* is a group which should be generated by a single element g, which is called *generator*. This group is an Abelian one.

$$\begin{cases} A = n_1 n_3 p_1 I_{n_2} \\ B = n_2 n_3 p_2 I_{n_1} \\ U = n_1 n_2 n_3 p_1 I_{p_2} \\ V = n_1 n_2 n_3 p_2 I_{p_1} \\ U' = n_2 n_3 p_1 I_{p_2} \\ V' = n_1 n_3 p_1 I_{p_1} \\ K_1 = \{0\} \\ K_2 = \{1,2, \dots n_3 - 1\} \end{cases} \quad (8)$$

All these conditions (8) permit to set all the parameters once we find the five numbers n_1, n_2, n_3, p_1, p_2.

Keep in mind that I_j is a set defined as $I_j = \{0,1, \dots j-1\}$; e.g. if $n_1 = j = 3$, than $I_{n_1} = \{0,1,2\}$.

Theorem 4. Let p_1, p_2 be different prime numbers, and n_1, n_2, n_3 be positive integers such that the product $n_1 p_1$ is coprime with $n_2 p_2$ and $n_1 p_1$ is coprime with $n_3 p_2$. Let H be the subgroup $H = p_2 I_{n_1 n_2 n_3}$ of Z_N with $N = n_1 n_2 n_3 p_1 p_2$, and K be a complete set of *cosets* [2] representatives for Z_N modulo H such that $K = K_1 \cup K_2$; then the pair (S, R) defined by

$$S = A + B \quad (9)$$

$$R = (U + V' + K_1) \cup (U' + V + K_2) \quad (10)$$

is a Vuza Canon of Z_N if R is a non-periodic subset of Z_N and:

$$\begin{cases} A = n_1 p_1 p_2 I_{n_2} \\ B = n_2 n_3 p_2 I_{n_1} \\ U = n_1 n_2 p_1 p_2 I_{n_3} \\ V = n_1 n_2 n_3 p_2 I_{p_1} \\ U' = n_2 p_2 I_{n_3} \\ V' = n_1 p_2 I_{p_1} \end{cases} \quad (11)$$

This last set (11) is the one we implemented in our algorithm.

4. Implementation of the Algorithm

In this section we discuss our algorithmic implementation of Vuza Canons [7]. We used a programming language called *Mathematica* developed by *Stephen Wolfram*, which has a powerful and intuitive management of lists and a myriad of already implemented features, some of which were fundamental to the implementation of this algorithm.

The program takes as input a value called *targetNumber* that establishes the maximum value within which good periods for Vuza Canons are generated. Then the good period list is created and displayed as output, the user chooses a good period, which is stored in a variable named *chosenPeriod,* the algorithm generates all the *quintuples* (the five numbers n_1, n_2, n_3, p_1, p_2) related to the chosen period, the user chooses the preferred quintuple, which is stored in a list called *chosenQuintuple*, and, finally, R and S are generated.

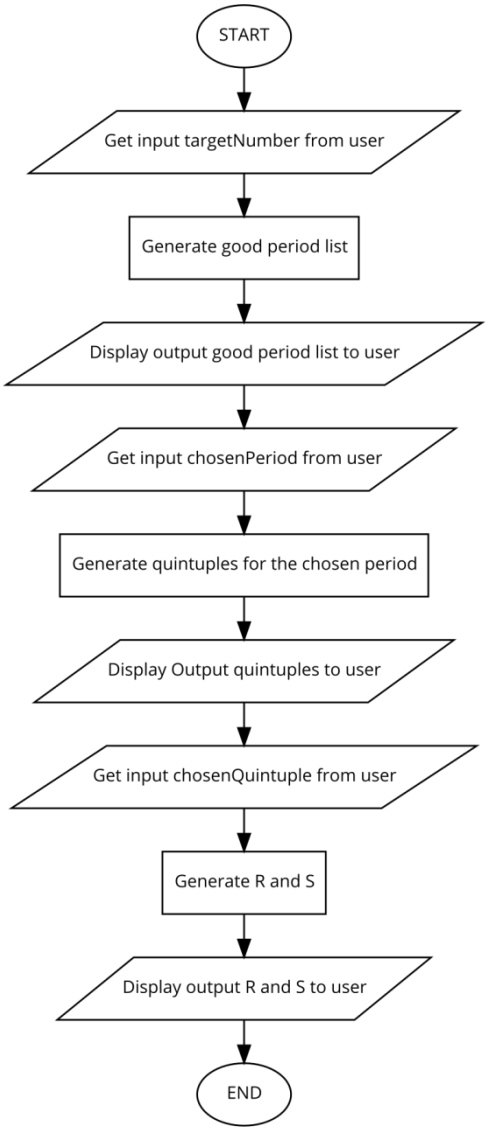

Figure 1. A concise flowchart of the algorithmic implementation

Below it is a detailed description of the algorithm.

The first step is to use a function we called *genPrimes* that generates prime numbers from 2 up to *targetNumber* (as clarified above, *targetNumber* is a user-specified value that influences the number of generated good periods). The generated prime numbers are stored in a list called *primeNumbers*.

Then, we know from *Theorem 1* that a period N (hereinafter «n») in order to be a good period must not satisfy the following conditions:

- $n = p^\alpha$ or $n = p^\alpha q$ with p and q prime numbers and α natural number between 1 and 10
- $n = pqr$ with p, q and r prime numbers
- $n = pqrs$ with p, q, r and s prime numbers
- $n = pqr^2$ or $n = p^2 q^2$

We need to create a list called *checkList*, which length is related to the length of *primeNumbers*, containing all the

2 If G is a group, and H is a subgroup of G, and g is an element of G, then $gH = \{gh : h \text{ an element of } H\}$ is the *left coset* of H in G with respect to g, and $Hg = \{hg : h \text{ an element of } H\}$ is the *right coset* of H in G with respect to g. For abelian groups, left cosets and right cosets are always the same.

periods that satisfy the above-mentioned conditions.

A function named *extractPeriods* analyzes a range of integers from 2 to *targetNumber* and returns only the integers that are not present in *checkList*; the returned values are all the good periods that are equal or less than *targetNumber*.

In the next part of the algorithm, the user chooses a good period, which is stored in *chosenPeriod* that must be factorized in such a way as to produce quintuples of integers.

To achieve the goal we use two functions called *FactorInteger* - natively implemented in *Mathematica* - and *extendedFact*.

The *FactorInteger* function gives a list of the prime factors of the provided integer, together with their exponents, for example *FactorInteger [12]* returns *{{2, 2}, {3, 1}}*. After that it is possible to transform the list *{base, exponent}* into a list that contains the individual factors without exponents by using the *extendedFact* function, for example *extendedFact [{2,3}]* returns *{2, 2, 2}*.

Then we apply the *extendedFact* function to all the sublists provided by the *FactorInteger* function: in the case of the good period $n = 72$ the *FactorInteger* function returns the list *{{2, 3}, {3, 2}}* and the function *extendedFact*, applied to the two sublists, returns *{2, 2, 2, 3, 3}*).

In order to define which elements of the quintuple are n_1, n_2, n_3, p_1, p_2, it is needed to check that $n_1 p_1$ is coprime with $n_2 p_2$ and that $n_1 p_1$ is coprime with $n_3 p_2$ using the *CoprimeQ* function, natively implemented in *Mathematica*.

In addition the following conditions must be met: $p_1 \neq p_2$, $n_1 p_1 \geq 2$, $n_2 p_2 \geq 2$ and $n_1 + n_2 + n_3 \geq 2$.

Applying the function to all possible permutations of the quintuple, the list of the possible good quintuples, sorted this way *{n1, p1, n2, p2, n3}*, will be returned.

At this point, the user chooses the preferred quintuple, which is stored in a list called *chosenQuintuple*, among those previously generated. After that it is possible to calculate S and then R (from equation 9 and 10).

Firstly, we need to generate A and B (denoted in the code as a and b) using the *Mathematica* natively implemented function called *Table* (*Table[expr,{i, 0, 5}]* generates a list of the values of *expr* when *i* runs from 0 to 5 with step 1):

```
a = Table[i * n1 * p1 * p2, {i, 0, (n2 - 1)}];
b = Table[i * n2 * n3 * p2, {i, 0, (n1 - 1)}];
```

The list named a has length n_2, while the list named b has length n_1; a is given by the multiplication of an i variable, ranging from 0 to $(n_2 - 1)$ multiplied by $n_1 p_1 p_2$. Something similar happens with the computation of b, but in this case i ranges from 0 to $(n_1 - 1)$ and is multiplied by $n_2 n_3 p_2$.

Then the additions of two values of each element of a with each element of b are calculated and stored in a list called *S*.

The previously chosen good period is appended at the end of the list and the list is flattened.

Now we need to calculate the differences between the elements contained in the *S* list using the *Mathematica* natively implemented function *Differences* in order to obtain S in its *basic form*, i.e.: if $S = \{\alpha, \beta, \gamma, \delta, \varepsilon, \vartheta\}$ with $\alpha, \beta, \gamma, \delta, \varepsilon, \vartheta \in Z_N$, then its basic form is given by $S_B = \{\beta - \alpha, \gamma - \beta, \delta - \gamma, \varepsilon - \delta, \vartheta - \varepsilon, N - \vartheta\}$. S represents the entry interval of each voice of the canon.

It is possible to calculate R using a similar strategy to the one used for S; R represents the rhythmic pattern of the single voice, a pattern that is repeated after a certain entry interval contained in the S list.

Now it is possible to calculate U, V, U', V' denoted in the code as $u, v, u1, v1$ as follows:

```
u = Table[i * n1 * n2 * p1 * p2, {i, 0, (n3 - 1)}];
v = Table[i * n1 * n2 * n3 * p2, {i, 0, (p1 - 1)}];
u1 = Table[i * n2 * p2, {i, 0, (n3 - 1)}];
v1 = Table[i * n1 * p2, {i, 0, (p1 - 1)}];
```

After that we define $K_1 = 1$ and $K_2 = 0$. Then it is necessary to compute all the possible additions of two elements between each element of $v1$ and u and to add $k1$. $u1vk2$ is then calculated in a similar way, except that the value of the constant $k2$ is equal to 0. After that the two resulting lists are joined, stored in a list called *R*, the chosen good period is appended and the list is flattened. In order to obtain R in the *basic form,* it is needed to apply the *Differences* function.

The algorithm is able to calculate all the good quintuples with maximum efficiency and then all the good R and S and it outputs R and S in the absolute and basic (or relative) form. It is also important to emphasize that S and R are interchangeable and that R can become S and vice versa.

5. Some Results

We generated both graphs and sounds. In this section, we will show some representations of a Vuza Canon with period 264 using the quintuple {11, 3, 2, 2, 2}.

In this case R in the basic form is *{1, 3, 19, 22, 43, 4, 41, 22, 21, 1, 3, 84}* and S in the basic form is *{8, 8, 8, 8, 8, 8, 8, 8, 2, 6, 2, 6, 2, 8, 8, 8, 8, 8, 8, 8, 8, 118}*.

In the following figures, it is possible to see the voices of the canon on the Y-axes and time, expressed in seconds, on the X-axes.

We are plotting a Vuza Canon, so, we obtain, as we should, only one point per X coordinate for each instant.

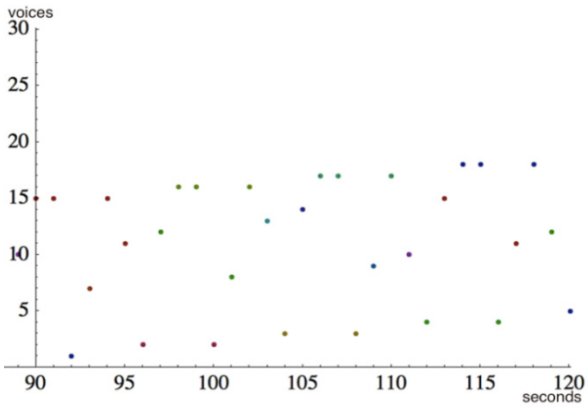

Figure 2. Eighteen voices of the Vuza canon with N=264.

In Figure 2, from ninety to one-hundred-twenty seconds, it is possible to see eighteen voices playing together.

In Figure 3, it is possible to see the end of the good period, 264, for the first voice of the canon.

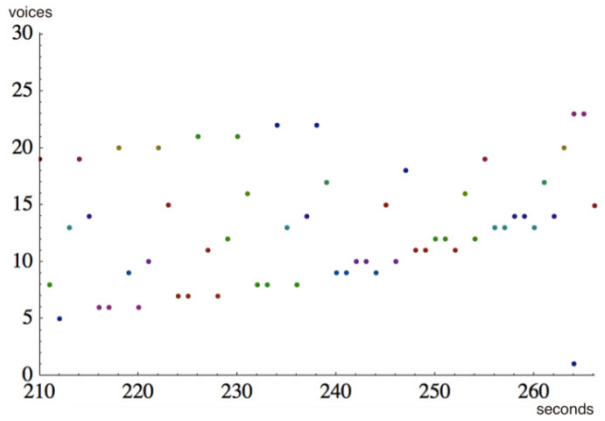

Figure 3. The first voice of the Vuza canon with N=264 at its end.

6. Real-time Sound Spatialization

New technologies offer many possibilities to composers working with the physical movement of sound, such as, for instance, trajectories followed by sound events through space, continuous modulation on harmonic and dynamic levels, and various types of proliferations of sound layers. As Berio asserted, «the most interesting possibilities are not these situations in themselves, but rather the relationships that are established between such physical–acoustic sound mobility and the effective mobility of the musical thought». (Giomi, F. / Meacci, M. / Schwoon, K. [8])

We agree that computer and sound diffusion technologies enable to live new, unusual acoustic spaces, making even the closed spaces versatile and accessible. This brings to the alteration of the listening perspective and to the creation of a multiplicity of sound levels in continuous transformation.

To reach this aim, we chose Vuza Canons to generate patterns for real-time sound spatialization. We have created a system that associates RCMC parameters to sound locations as sequences of configurations of loudspeakers. It consists of a series of engines that operate on the amplitude of the audio signal, redirecting it toward a variable number of outputs according to the data obtained thanks to the *Mathematica* implementation shown above.

We use the Max/MSP environment to manage the reproduction of sound and its spatialization. As we hear sound in the form of power, we have chosen to use the equal power panning law, which means that we perceive the dynamic level of audio signals as the square of the input amplitude. To obtain an equal and smooth panning, a linear one, it is necessary to use the inverse of the square operator, which is the square root. Obviously, the square root is not a linear function but the important point is that we perceive the operation imposed on the signal to be linear, as the square is neutralized by the square root, rendering a linearly changing pan when turning the pan pot linearly.

The equations we implemented are:

$$y[n]_{speaker1} = \sqrt{x[n]} \tag{10}$$

$$y[n]_{speaker2} = \sqrt{1.0 - x[n]} \tag{11}$$

where $y[n]$ is the amplitude of the considered speaker of $x[n]$ input signal.

We applied this algorithm to an octophonic sound system. R and S are used to define the active speaker and the transition time from one speaker to another.

The Max/MSP code reads the selected R and S from a text file as shown in figure 4. The *counter* object keeps counting using bang messages to read each line of the file. An example of the output file produced by Mathematica is shown in figure 5.

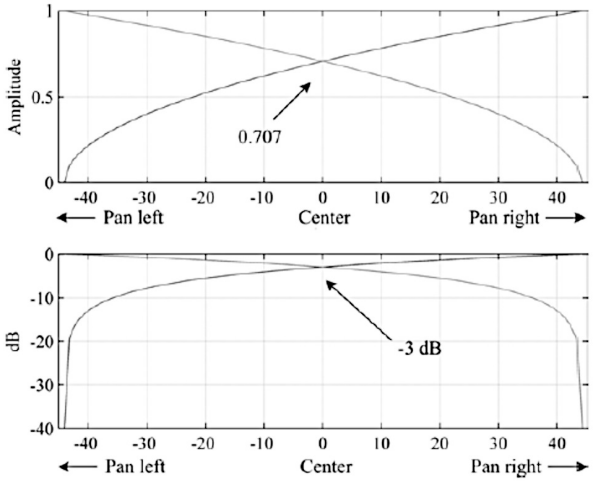

Figure 4. Equal power panning with root of the amplitude (top) and dB of the root amplitude (bottom). Image from "Introduction to digital signal processing" by Tae Hong Park – World Scientific Press 2009.

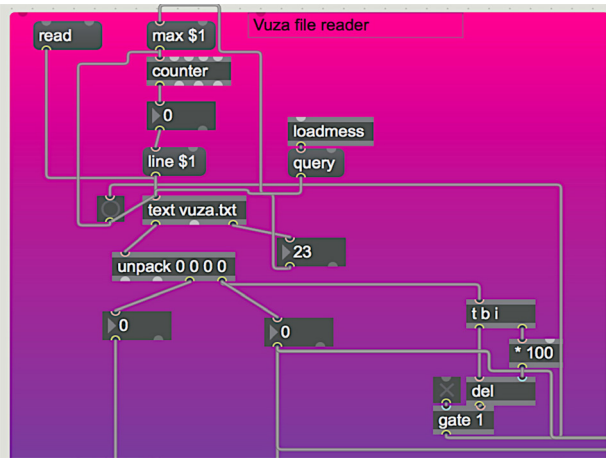

Figure 5. From the Max/MSP environment: the code reads R and S from a text file

Then, the R and S array elements are scaled, if necessary, and sent to the *line_sqrt* subpatch that realizes the equal panning applied to all the possible transitions between the eight speakers.

```
1 0 0
2 1 8
3 4 16
4 23 24
5 45 32
6 88 40
7 92 48
8 133 56
9 155 64
10 176 66
11 177 72
12 180 74
13 264 80
14 0 82
15 1 90
16 4 98
17 23 106
18 45 114
19 88 122
20 92 130
21 133 138
22 155 146
23 176 264
```

Figure 6. An extract from the output text file generated by Mathematica

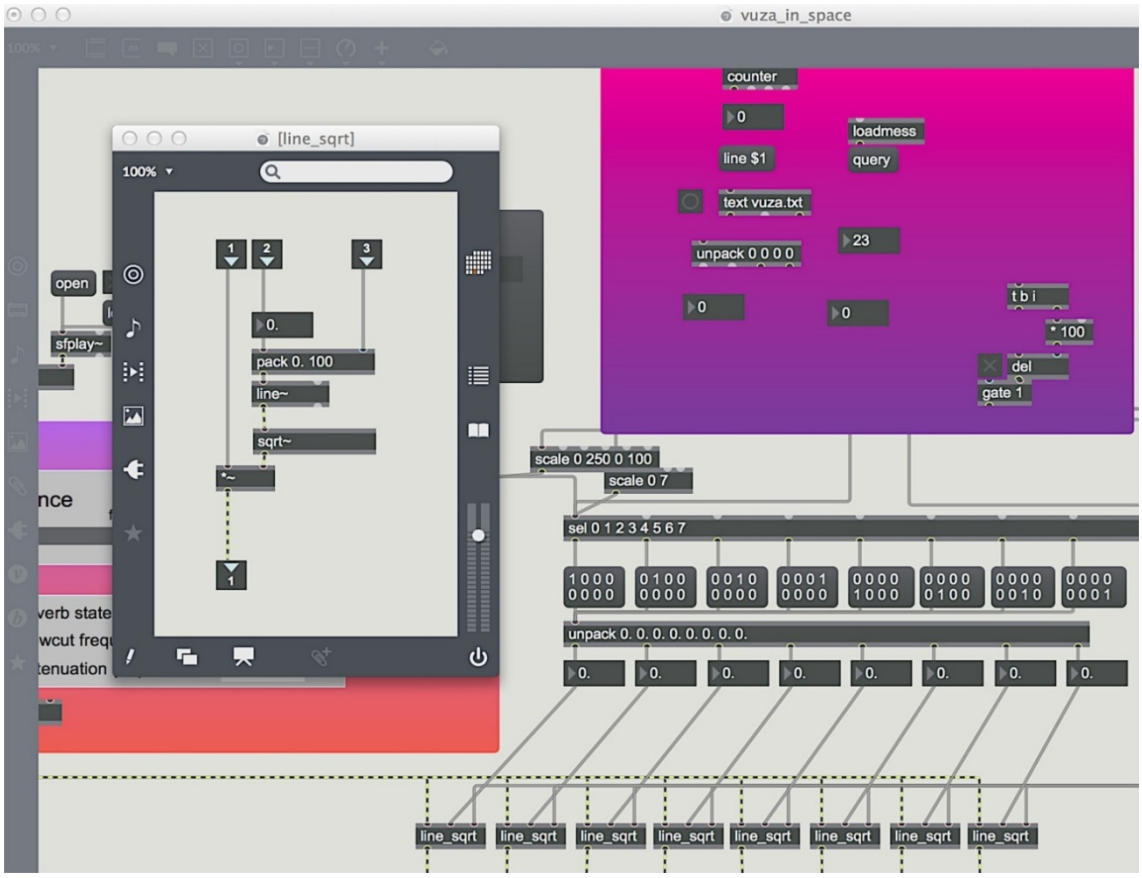

Figure 7. A screenshot of the Max/MSP environment. On the left side the implementation of the equal power panning law.

7. about VUZALIZER

In order to simplify the use of RCMC Canons for all users, we created a Max/MSP object that returns real-time output of the R and S sets.

This Max object is built around the previously described algorithm. Using this object Max/MSP users can easily calculate S and R, for the chosen period, and employ them in their works.

Our project is made up of the following files:

vuzalizer.maxpat
vuzalizer_logic.maxpat
vuzalizer_logic.js
vuzalizer.db

We will describe them in order of importance.

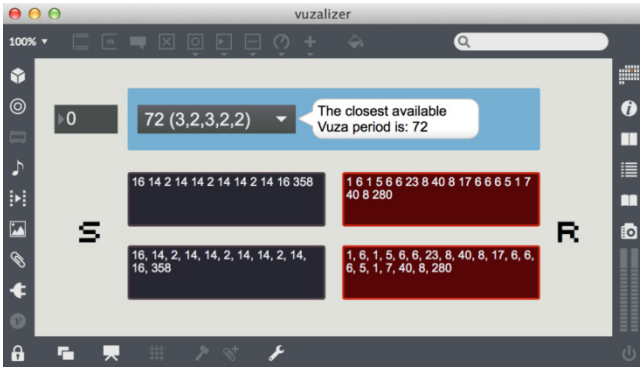

Figure 8. A screenshot of the Max/MSP environment. The *vuzalizer* object returns lists of S and R for the chosen period.

7.1. vuzalizer.db

This is the core of the object: a SQLite database containing all the good periods - and the corresponding quintuples - up to 1000 calculated by our algorithm, but we are going to enlarge it in the near future up to 10000 possible canons.

7.2. vuzalizer_logic.js

This javascript code takes care of the interaction with the database; we need to use javascript to communicate between Max/MSP and a SQLite database. The code locates the database file on the disk and queries it to extract the data requested by the Max/MSP user. Our object reads data from the database.

7.3. vuzalizer_logic.maxpat

It contains each element of the UI and it takes care of managing the connections between the javascript, the UI, the inlet and the outlet. It calls vuzalizer_logic.js to query the database to populate the umenu. It processes all the data it receives to fill the different elements of the UI and format them as zl lists, CSV strings or Max/MSP messages. It also supports a method to choose the closest good period to a user-input number.

7.4. vuzalizer.maxpat

This is an example of use of the vuzalizer_logic.maxpat inserted as a bpatch (figure 7). It is possible to see the umenu already populated with all the good periods (and the corresponding quintuple) found in the database. By choosing a menu item, the object will output both of the S and R strings in the form of two zl lists, two comma separated value strings or two sequences of Max/MSP messages. Furthermore it is possible to give as input any number in the inlet and Vuzalizer will choose the closest good period for the user.

8. Conclusions

Regular Complementary Canons of Maximal Category were presented to the musical world in the 1990s and they are still a relevant topic of research. Nowadays RCMC Canons can be implemented using a programming language, allowing composers to use them with ease.

The study of Vuza Canons is still an open field of research in the domain of formalization and implementation of new musical structures.

In this paper we introduced the RCMC Canons and discussed our algorithmic implementation, then we used Vuza Canons to generate patterns for real-time sound spatialization and, finally, we implemented a Max/MSP object that returns real-time output of the R and S sets. We intend to continue our research in order to gain insight into modulations between different Vuza Canons.

Acknowledgements

Mathematica fonts by Wolfram Research, Inc.

REFERENCES

[1] Fidanza, G. (2008). Canoni ritmici a mosaico. Master's degree thesis

[2] Szabo, S. and Sands, A. D. (2009) Factoring Groups into Subsets. Boca Raton, CRC Press, 2009

[3] Vuza, D. T. (1991). Supplementary sets and regular complementary unending canons. Perspectives of New Music.

[4] Brujin, N. G. (1953). On the factorization of cyclic groups. Indag. Math. 15, pp. 258-264.

[5] Amiot, E. (2011). Structures, algorithms and algebraic tools for rhythmic canons. Perspectives of new music, Vol. 49 No. 2, Summer 2011

[6] Jedrzejewski, F. (2013). On the enumeration of Vuza canons.

[7] Battista, A. F. / Mollo, C. M. / Monopoli, N. (2015). Rcmc canons: not only a problem of cage. Proceedings of

ICMC2015, ISBN 10-0- 9845274-4-3, 2015

[8] Giomi, F. / Meacci, M. / Schwoon, K. (2003) Live electronics in Luciano Berio's music, Computer Music Journal 27 (2), MIT Press – pp. 30-46

[9] Amiot, E. (2004) Why Rhythmic Canons are Interesting, in E. Lluis-Puebla, G. Mazzola et T. Noll (eds.), Perspectives of Mathematical and Computer-Aided Music Theory, EpOs, 190–209, Universität Osnabrück

[10] Amiot, E. (2008) Autosimilar Melodies, Journal of Mathematics and Music, July, vol. 2, n◦ 3, 157-180

[11] Amiot, E. (2005) Rhythmic canons and Galois Theory, Grazer Math. Ber., 347, 1–25.

[12] Amiot, E. (2009) New perspectives on rhythmic canons and the spectral conjecture , in Special Issue "Tiling Problems in Music", Journal of Mathematics and Music, July, vol. 3, n◦ 2

[13] Andreatta, M. (2004) On group-theoretical methods applied to music: some compositional and implementational aspects, in E. Lluis-Puebla, G. Mazzola et T. Noll (eds.), Perspectives of Mathematical and Computer-Aided Music Theory, EpOs, 122–162, Universität Osnabrück

[14] Andreatta, M. (2007) De la conjecture de Minkowski aux canons rythmiques mosaïques, L'Ouvert, n◦ 114, p. 51-61, march

[15] Coven, E., and Meyerowitz, A. (1999) Tiling the integers with one finite set, in: J. Alg. (212), 161-174

[16] Davalan, J.P. (2011) Perfect rhythmic tilings, PNM

[17] DeBruijn, N.G. (1956) On Number Systems, Nieuw. Arch. Wisk. (3) 4, 15–17

[18] Fripertinger, H. (2005) Remarks on Rhythmical Canons, Grazer Math. Ber., 347, 55–68.

[19] Hajós, G. (1954) Sur les factorisations des groupes abéliens, in: Casopsis Pest. Mat. Fys. (74), 157-162.

[20] Hall, R., Klinsberg, P. (2006) Asymmetric Rhythms and Tiling Canons, American Mathematical Monthly, Volume 113, Number 10, 887-896.

[21] Johnson, T. (2001) Tiling The Line, proceedings of J.I.M., Royan

[22] Jedrzejewski, F. (2004) A simple way to compute Vuza canons, MaMuX seminar

[23] Kolountzakis, M. (2003) Translational Tilings of the Integers with Long Periods, Elec. J. of Combinatorics 10(1), R22

[24] Laba, I. (2002) The spectral set conjecture and multiplicative properties of roots of polynomials, J. London Math. Soc. 65, 661-671

[25] Newman, D.J. (1977) Tesselation of Integers, J. Numb. THeory 9, 107-11

[26] Rahn, J. (1980) Basic Atonal Theory, New York, Longman

[27] Steinberger, J.P. (2009) Tilings of the integers can have superpolynomial periods, Combinatorica, 29, 503-509.

[28] Steinberger, J.P. (2005) Multiple tilings of Z with long periods, and tiles with many-generated level semigroups, New York Journal of Mathematics, 11, 445-456.

[29] Swenson, C. (1974) Direct sum subset decompositions of Z, Pacific J. Math. 53, 1974, 629-633

[30] Tangian, A. (2003) Constructing Rhythmic Canons, PNM, Vol. 41, no. 2, 66–95

[31] Tijdeman, R. (1995) Decomposition of the Integers as a direct sum of two subsets, in: S´eminaire de th´eorie des nombres de Paris, 3D, Cambridge U.P, 261-276

[32] Szabò, S. (1985) A type of factorization of finite abelian groups, Discrete Math. 54, 121–124

[33] Warusfel (1971) Structures Algébriques finies, Classiques Hachette

[34] Wild, J. (2002) Tessellating the chromatic, Perspectives of New Music